Training robust neural networks

14/12/2021

Christophe - Guillaume - Yoann

Adversarial Defense Mecanisms

1) Adversarial Training

We trained a classifier on CIFAR-10 images that are adversarially attacked.
We use the following **PGD-Linf** attack:

- 10 iterations
- Epsilon = 10/255
- Step size = 2/255

$$\min_{\theta} \mathbb{E}_{(x,y)} \left(\max_{||\tau|| \leq \epsilon} L_{\theta}(x + \tau, y) \right)$$

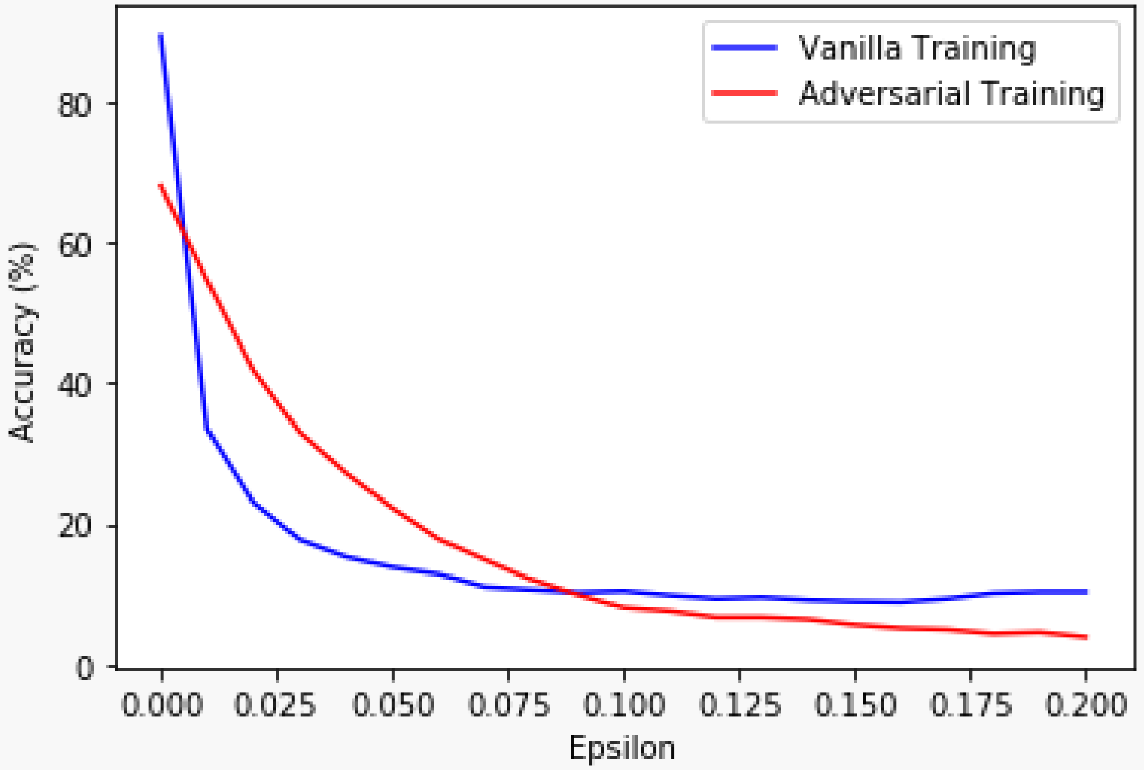
We trained two models (vanilla & adversarial training) with the following hyperparameters :

- 100 epochs
- Cosine Annealing Learning Rate : 0.1 -> 0
- Data augmentation : Random Horizontal Flip & Cropping

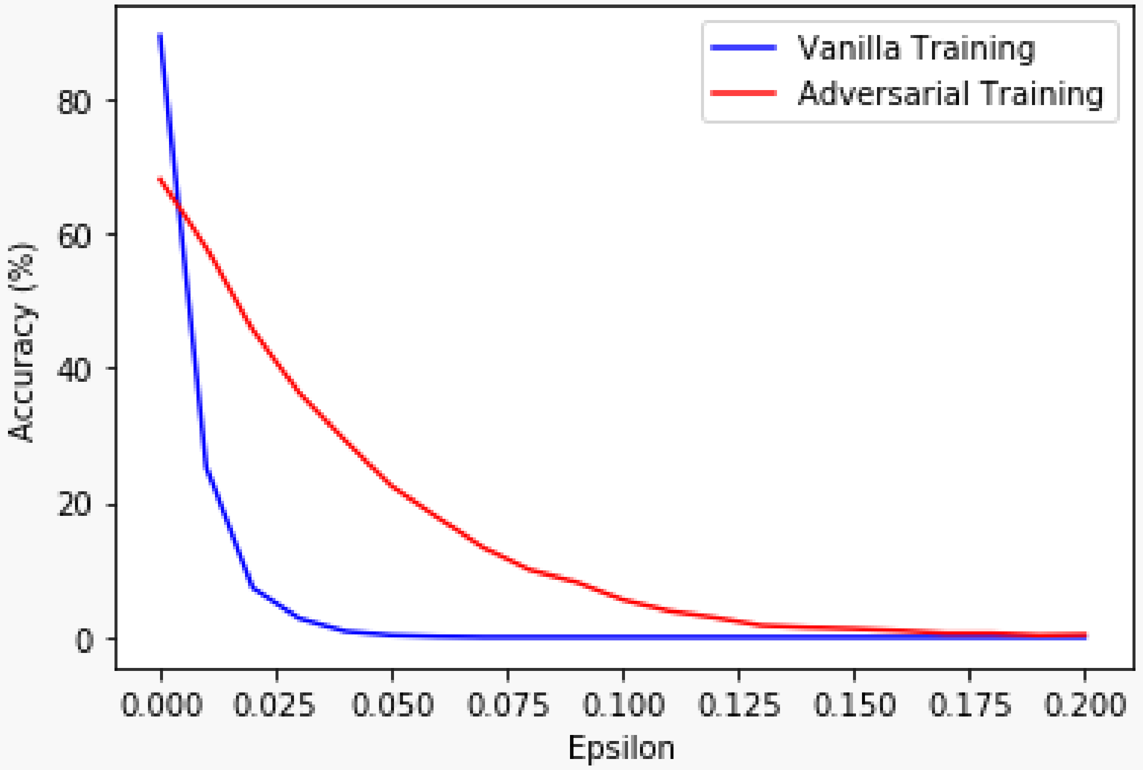
1) Adversarial Training

10 iterations
stepsize = epsilon / 10

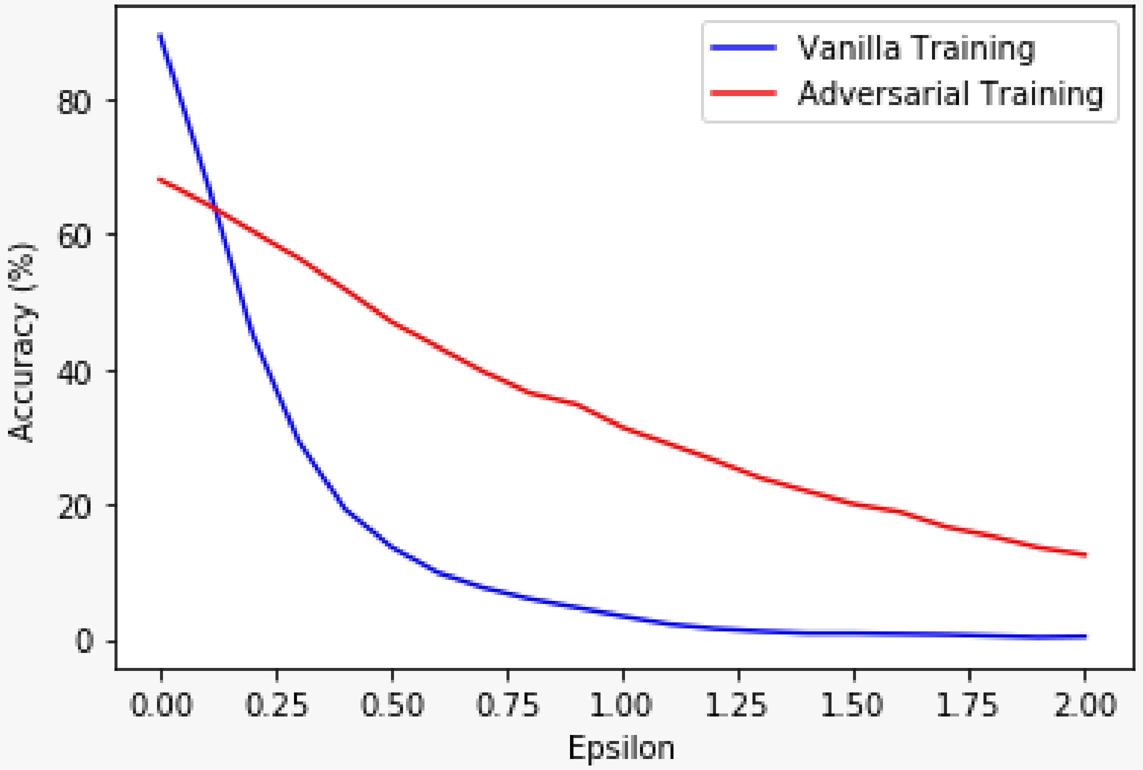
FGSM



PGD Linf



PGD L2

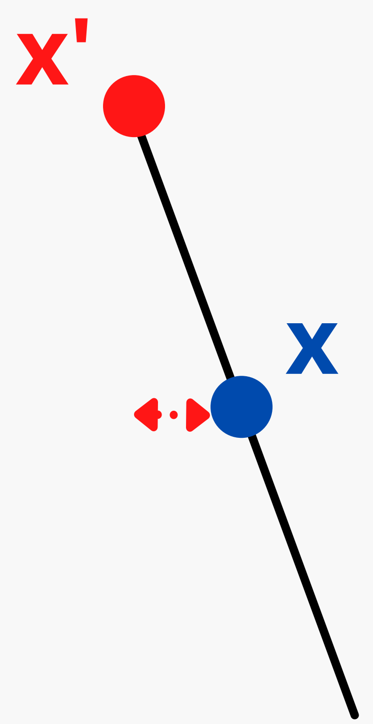


2) Idea : Gradient Norm Minimization

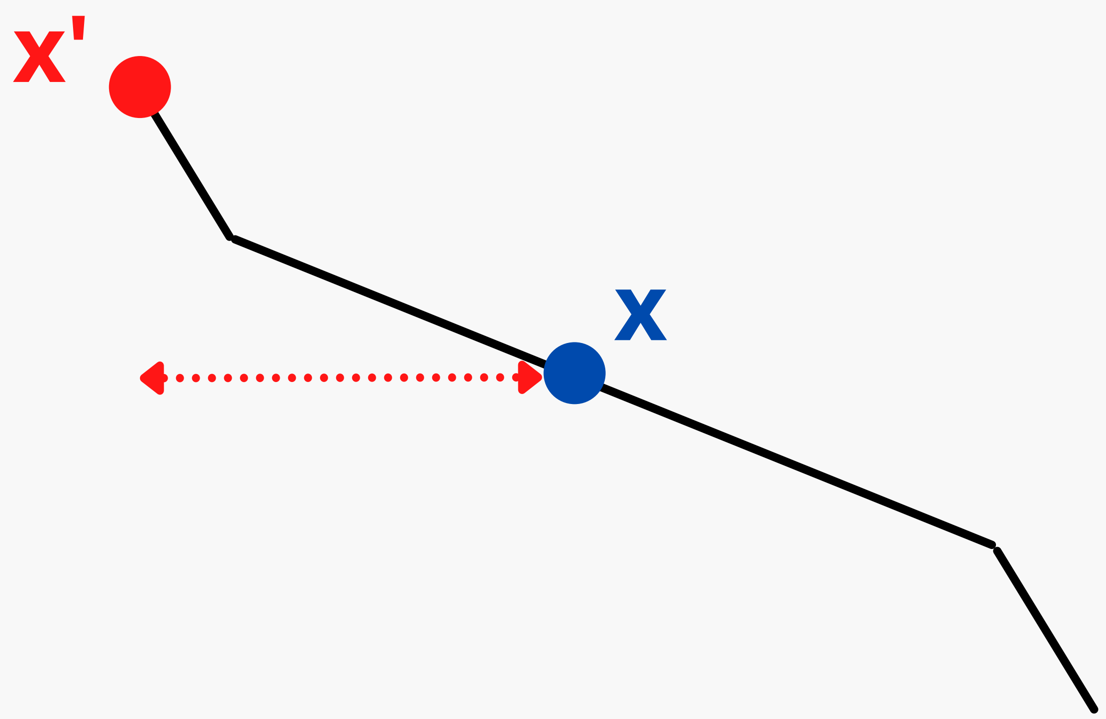
We added a **constraint term** to the loss function in order to **minimize the L2 norm of the loss gradients w.r.t. the inputs pixels.**

We trained such a model using the same training hyperparameters as before.

$$\text{Loss}_{GNM}(x,y) = L(x,y) + \lambda * \| \Delta_x L(x,y) \|_2$$



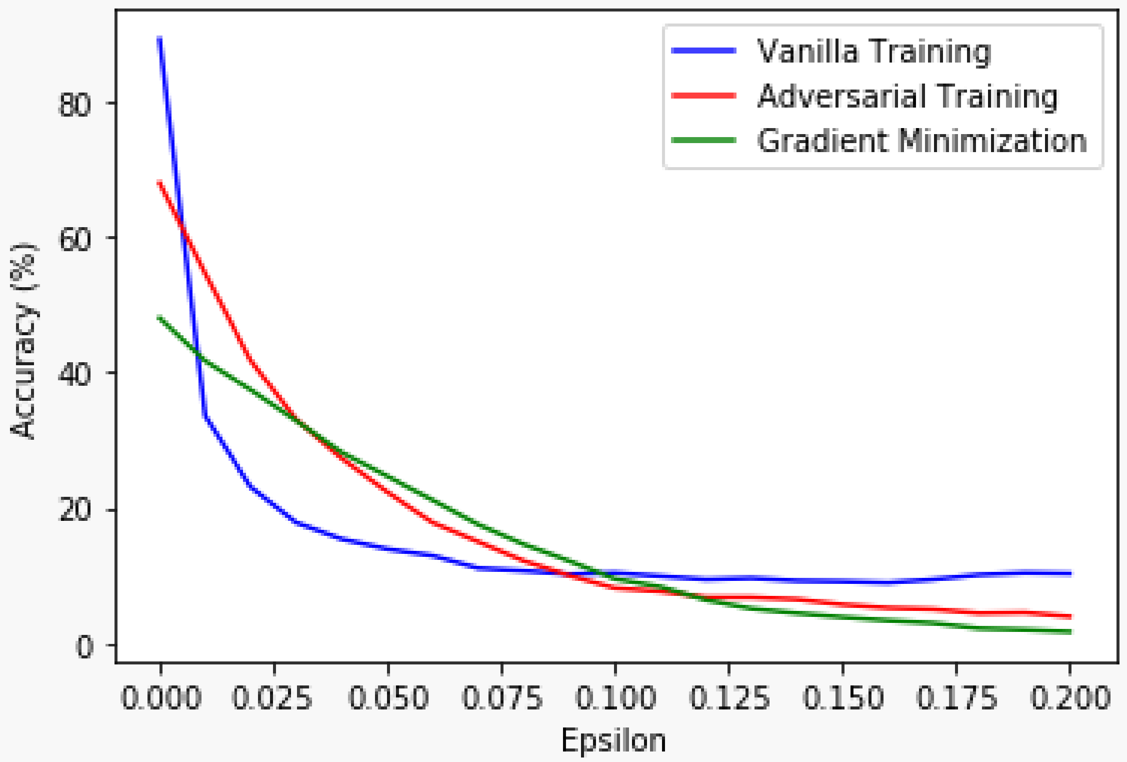
x' is "near" = invisible perturbation



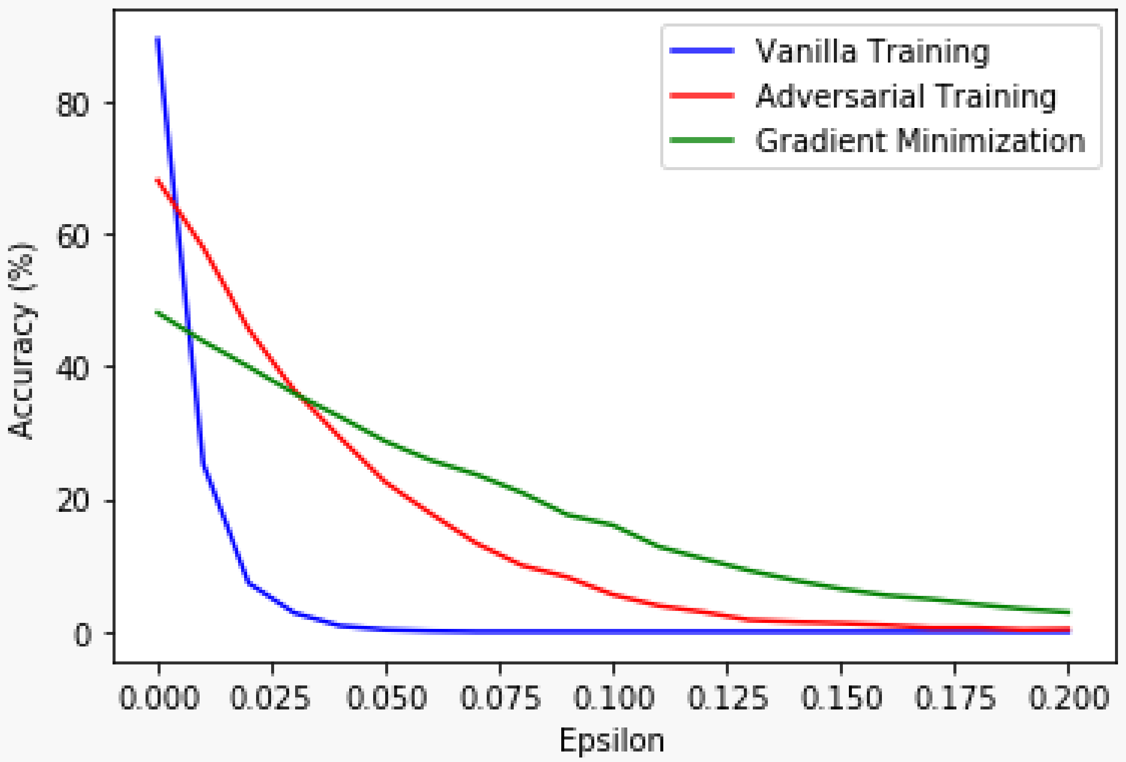
x' is "far" = visible perturbation

2) Idea : Gradient Norm Minimization

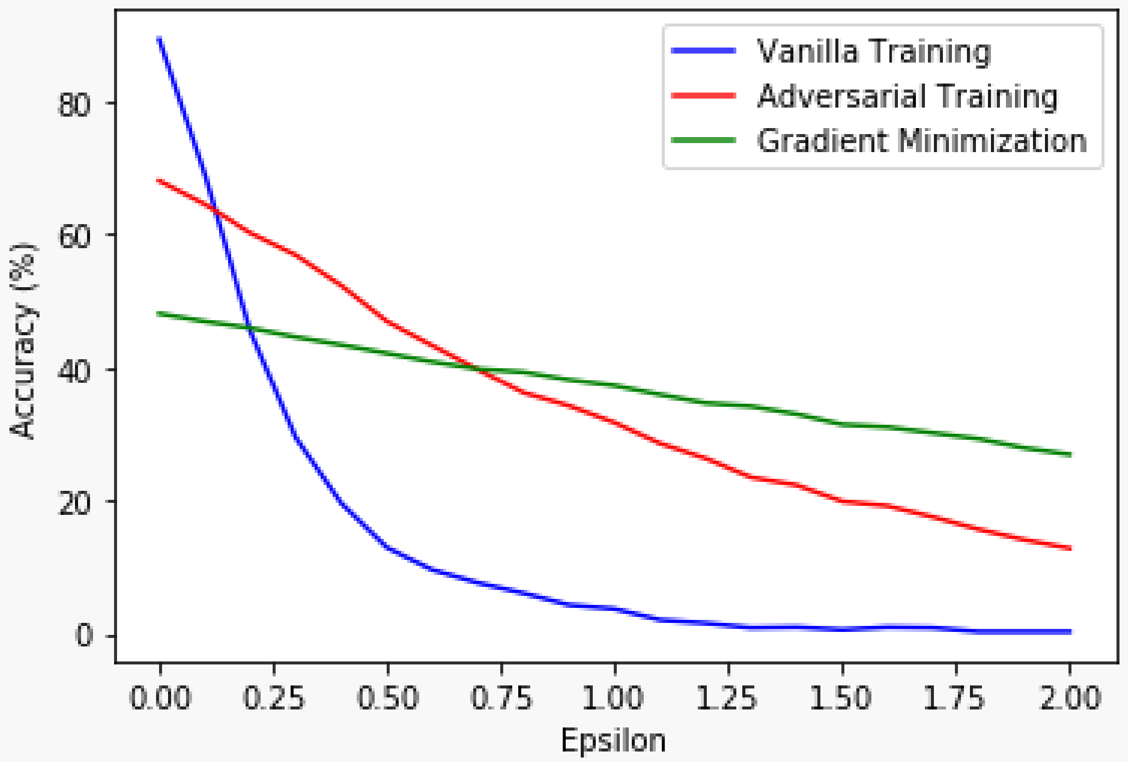
FGSM



PGD Linf



PGD L2



Our idea is an instance of **Gradient obfuscation**: making the gradients small / noisy to confuse gradient-based attacks.

It has been shown to be ineffective against adaptative attacks :
"Obfuscated Gradients Give a False Sense of Security", Anish Athalye, Nicholas Carlini, David Wagner, ICML 2018.

3) Randomized Networks

Injecting noise at inference time improves robustness of the network.
Results for Gaussian noise for a classifier on CIFAR-10 with 73% accuracy on validation :

- FGSM ($\varepsilon = 0.025$) : **~2%** acc (without noise), **~10,7%** acc (with noise, std = 0.1), **~16%** acc (with noise, std = 0.25)
- PGD L ∞ ($\varepsilon = 0.01$, iterations = 10) : **~13%** acc (without noise), **~33%** (with noise std = 0.25)
- PGD L ∞ ($\varepsilon = 0.025$, iterations = 10) : **~0.07%** acc (without noise), **~12%** (with noise, std = 0.1), **~16%** (std = 0.25).

Injecting noise during training at selected layers also improves robustness, but slightly decreases accuracy for normal examples.

4) Autoencoders as adversarial defense

We used two autoencoders as described in the paper presenting the MagNet defense :

- one as "detector" : it tries to approximate the manifold of normal examples.
- one as "reformer" : it pushes the adversarial examples to be close to the approximated manifold.

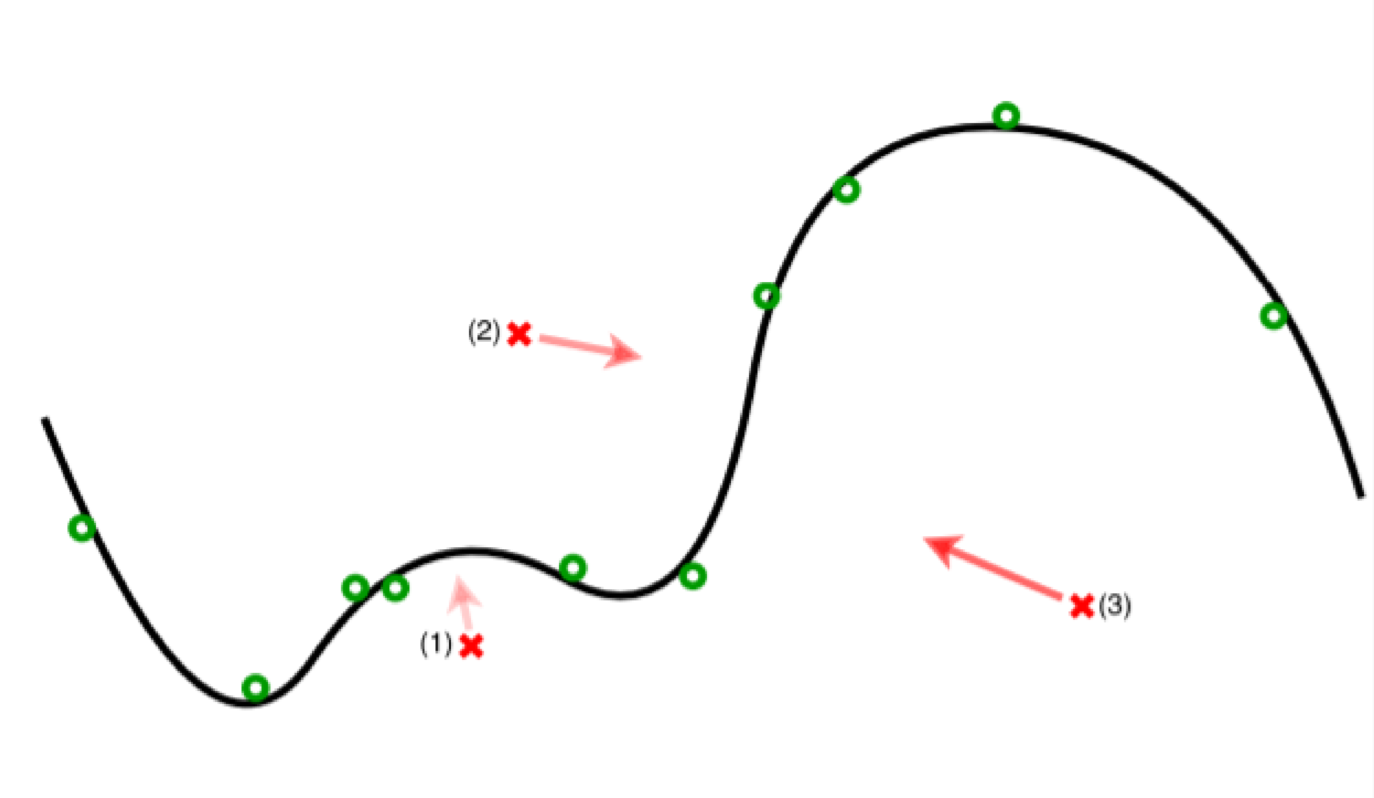
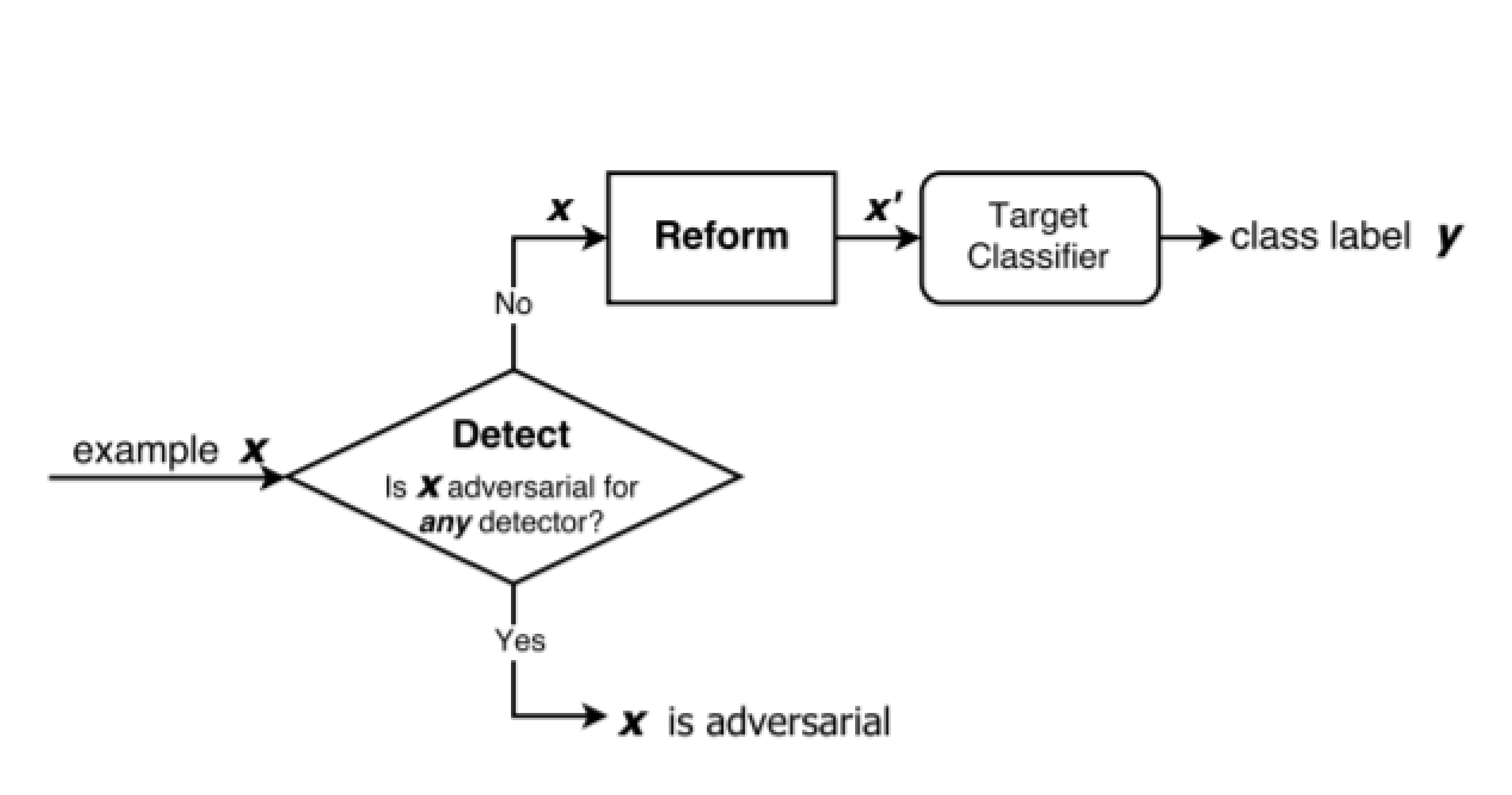


Illustration of how detector and reformer work in a 2-D sample space

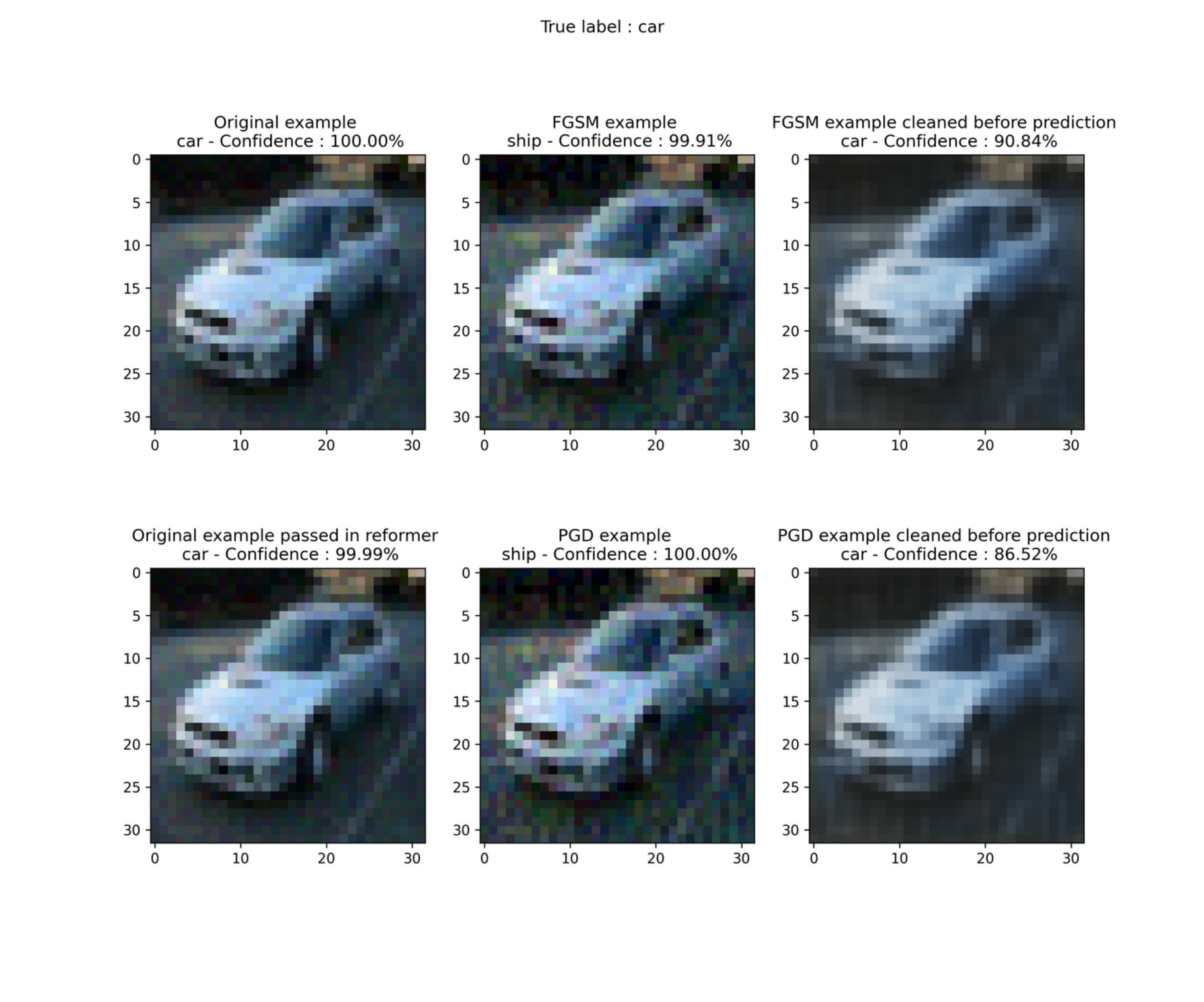


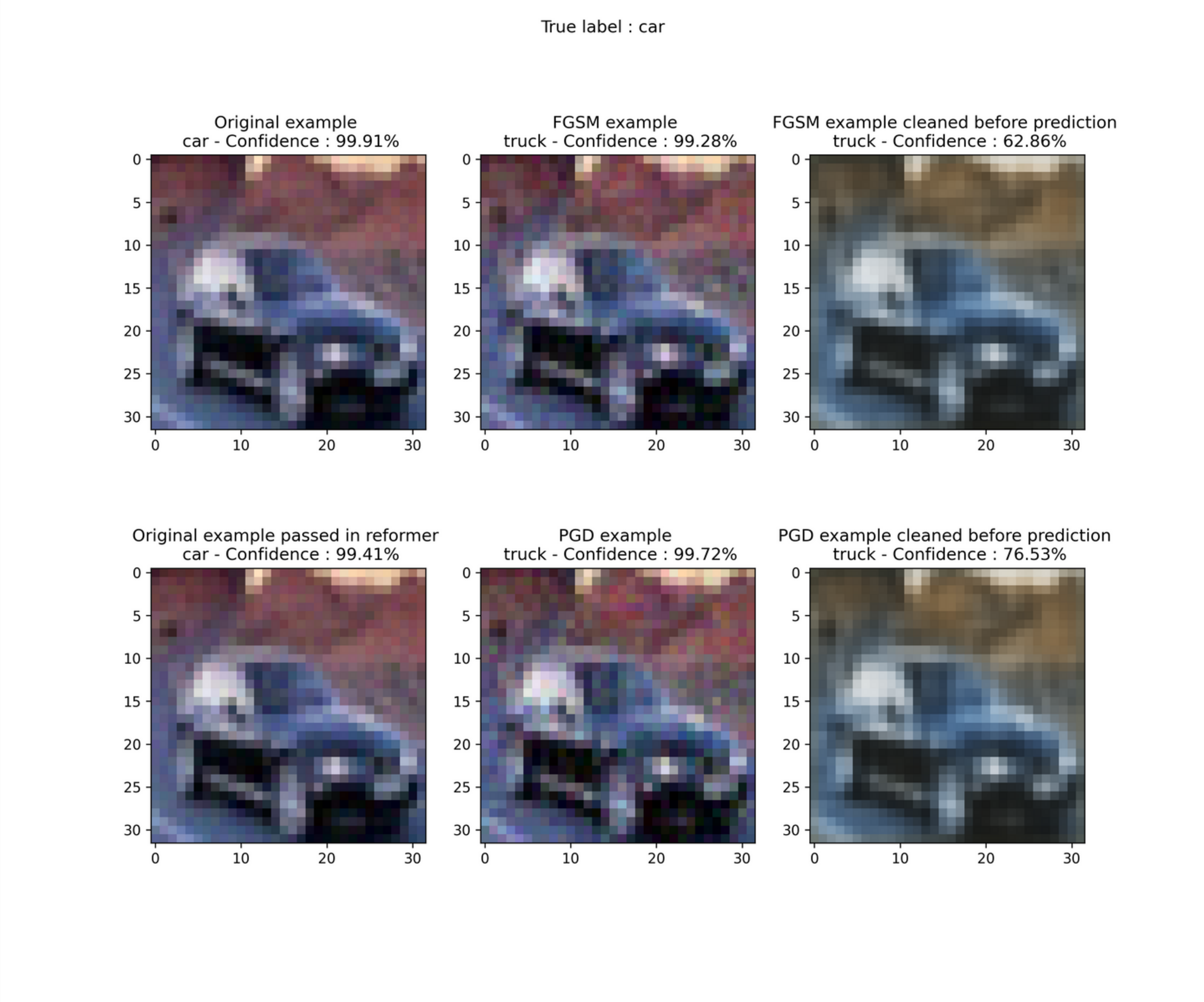
Workflow of MagNet

Source : *MagNet: a Two-Pronged Defense against Adversarial Examples, Dongyu Meng, Hao Chen* (2017)

4) Autoencoders as adversarial defense

Results obtained for slightly trained autoencoders (~10 epochs) :





4) Autoencoders as adversarial defense

Results obtained for slightly trained autoencoders (~10 epochs, ~70% acc on validation) for a classifier on CIFAR-10 with **73%** accuracy on validation :

- Without attack, with autoencoders : **~67%** acc.
- FGSM (ε = 0.01) : **~22%** acc (without AE), **~39%** acc (with AE)
- FGSM (ε = 0.025) : **~4%** acc (without AE), **~18%** acc (with AE)
- FGSM (ε = 0.05) : ~**0.4%** acc (without AE), ~**3.9%** acc (with AE)
- PGD L ∞ (ε = 0.01, iterations = 10) : **~13%** acc (without AE), **~39%** (with AE)
- PGD L ∞ (ε = 0.025, iterations = 10) : **~0.07%** acc (without AE), **~10%** (with AE)

Black-box Adversarial Attacks

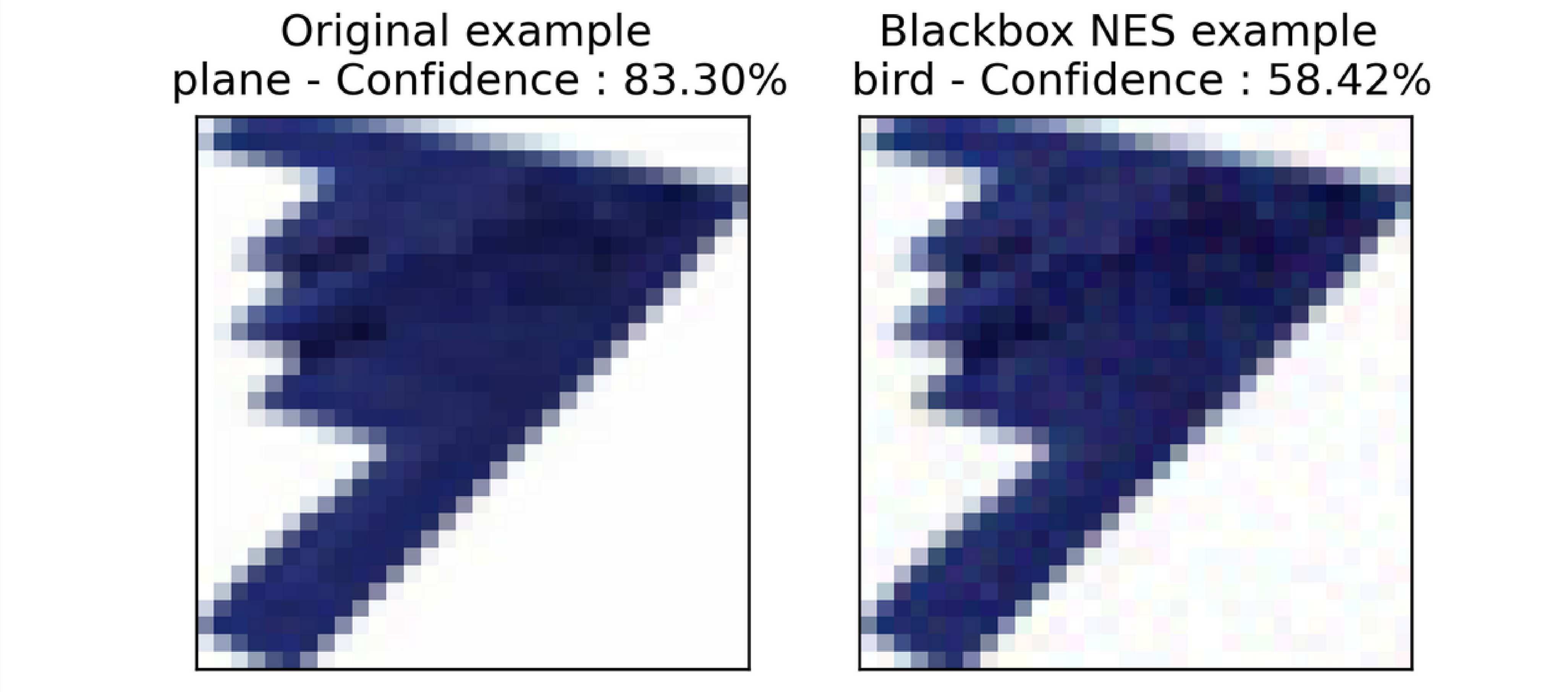
1) Query-limited setting

- Available information :

plane : 83%, **bird** : 16%, **ship** : 1%, **car** : 0%, **cat** : 0%, **deer** : 0%, **dog** : 0%, **frog** : 0%, **horse** 0 %, , **truck** : 0%

- Idea of the attack :

Estimate the gradient using **NES algorithm.**
Generate adversarial example using **PGD** with the estimated gradient.



Time for 1 attack :
~1 sec

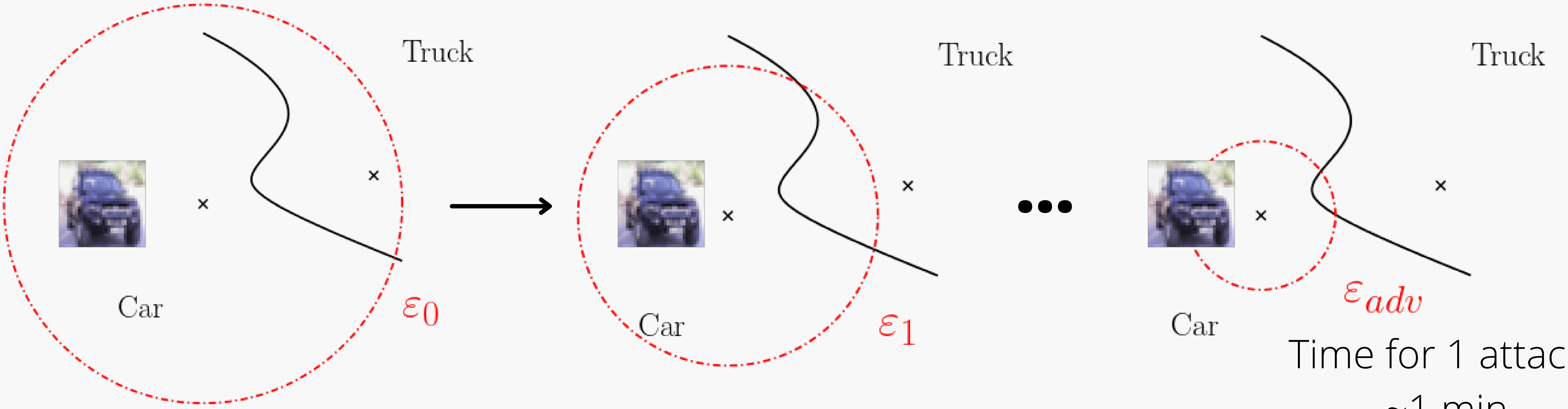
2) Partial-information setting

- Available information :

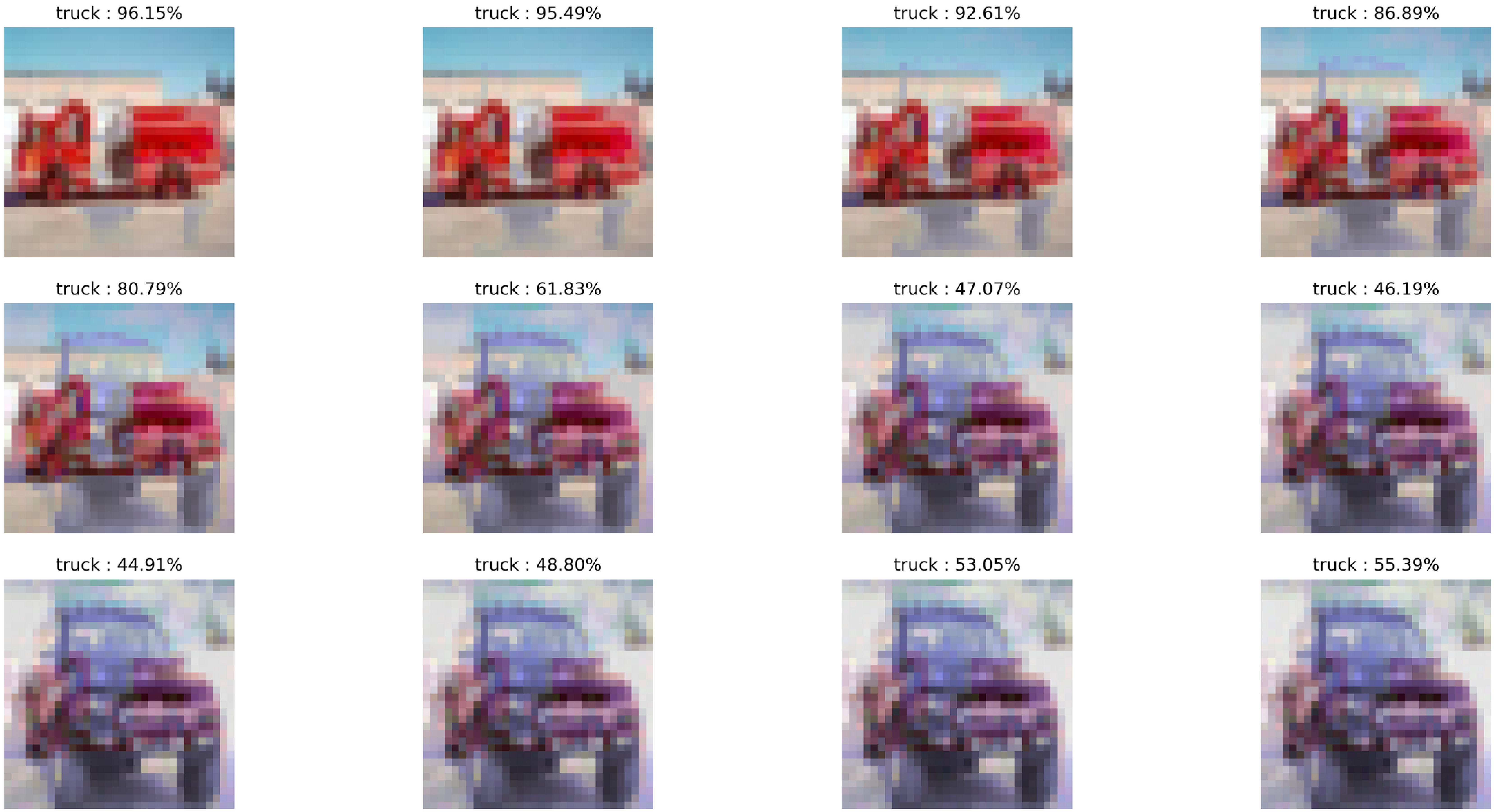
truck : 83%, **car** : 0%, **bird** : 16%, **cat** : 0%, **deer** : 0%, **dog** : 0%, **frog** : 0%, **horse** 0 %, **ship** : 1%, **truck** : 0%

- Idea of the attack :

Target an adversarial class, start with big **ε** so target class appear in top prediction, iteratively reduce ε while kipping **top prediction = target.**



Time for 1 attack : ~1 min


truck : 96.15%
truck : 95.49%
truck : 92.61%
truck : 86.89%
truck : 80.79%
truck : 61.83%
truck : 47.07%
truck : 46.19%
truck : 44.91%
truck : 48.80%
truck : 53.05%
truck : 55.39%

3) Label-only setting

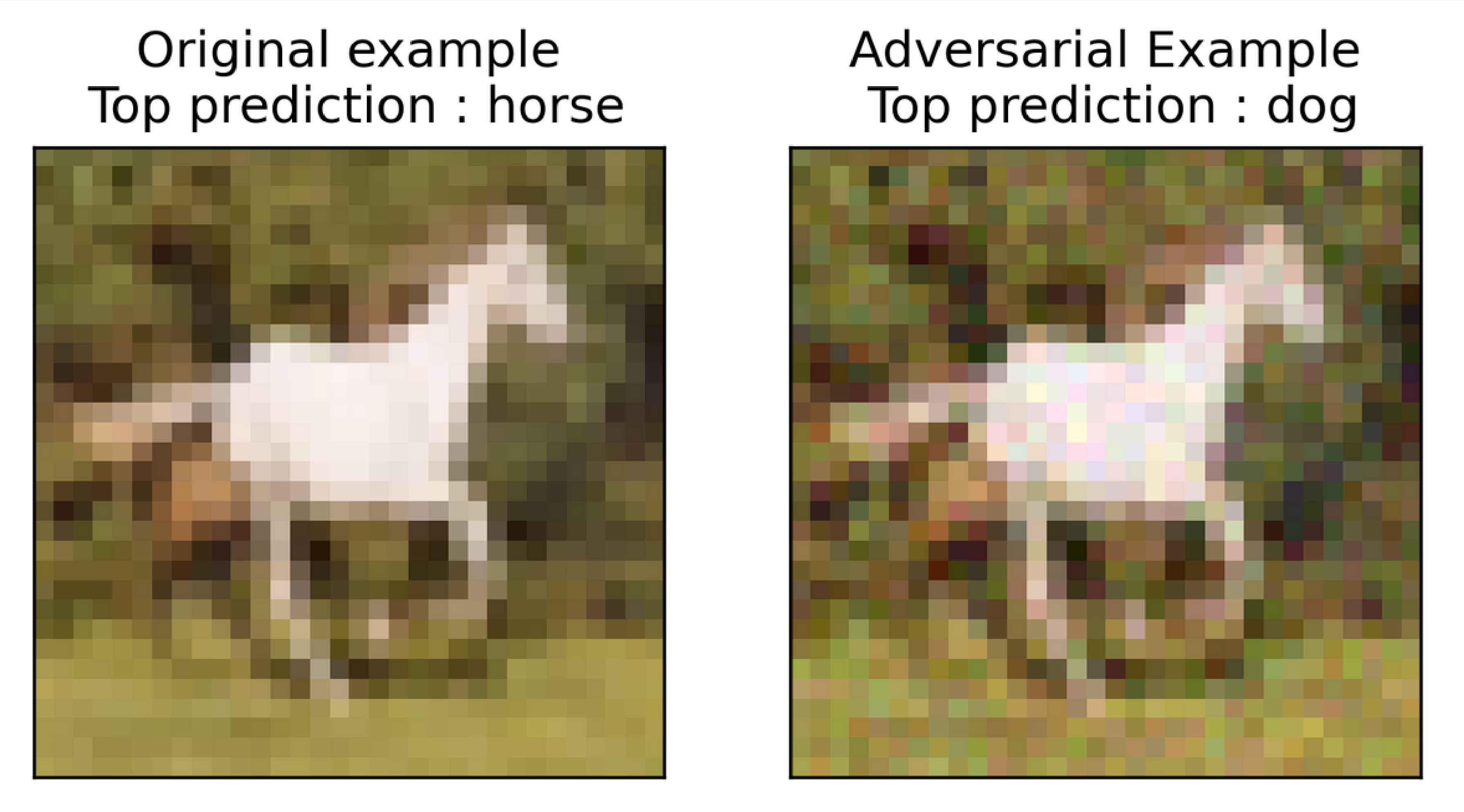
- Available information :

truck : 83%, **car** : 0%, **bird** : 16%, **cat** : 0%, **deer** : 0%, **dog** : 0%, **frog** : 0%, **horse** 0 %, **ship** : 1%, **truck** : 0%

- Idea of the attack :

Estimate a proxy score using random perturbations :

$$\widehat{S}(x^{(t)}) = \frac{1}{n}\sum_{i=1}^{n} R(x^{(t)} + \mu\delta_i)$$



Time for 1 attack :
~30 min

Any questions ?